Optimization of Mean-Shift scale parameters on the EGEE grid

Ting LI, Sorina CAMARASU-POP, Tristan GLATARD, Thomas GRENIER, Hugues BENOIT-CATTIN
Université de Lyon, CNRS, INSERM, CREATIS, 69621 Villeurbanne, FRANCE

Abstract. This paper studies the optimization of Mean-Shift (MS) image filtering scale parameters. A parameter sweep experiment representing 164 days of CPU is performed on the EGEE grid. The mathematical foundations of Mean-Shift and the grid environment used for the deployment are described in details. The experiments and results are then discussed highlighting the efficiency of gradient ascent algorithm for MS parameters optimization and a number of grid observations related to data transfers, reliability, task scheduling, CPU time and usability.

Keywords. Mean-Shift image filtering, scale parameters optimization, EGEE grid.

1. Introduction

Medical studies increasingly use multi-modality imaging, producing multidimensional data that bring additional information but that are also challenging to process and interpret. They need advanced algorithms such as multidimensional filtering, segmentation and clustering to get the really useful information out of them and to discover new knowledge.

Based on the feature space analysis and on non-parametric kernel estimation, Mean-Shift is a broadly used robust method to address these problems. It was introduced by Fukunaga [1] in 1975 and firstly used for image filtering by Comaniciu in 1997 [2]. Alternate methods include the one described in [3].

Many papers have illustrated the Mean-Shift robustness [4,5] and its superiority over classical filtering methods such as anisotropic diffusion and bilateral filtering [6]. Mean-shift was also successfully applied to medical image segmentation and filtering [7,8,9]. The large success of Mean-Shift is mainly due to the intuitive tuning of parameters offered by the approach [10]. These parameters describe the scale at which features are analyzed and have a strong influence on the results quality. However, without prior knowledge, no rule is known to determine the optimal values of scale parameters (for statistical approaches refer to [11,12], using the impulse response of the acquisition system see [13]).

In this work we propose to determine whether Mean-Shift scale parameters can be optimized using gradient ascent. Robust convergence of the gradient ascent to the global maximum requires that the multidimensional parameter space has only one basin of attraction, i.e., that the objective criterion used in the optimization has no local maximum. The experiment presented here intends to test it through an exhaustive search in the pa-

rameter space. A noisy image is filtered with numerous different scale parameters and the filtering quality is then quantified with the signal-to-noise ratio. Given the number of parameter combinations involved this is hardly achievable with a state-of-the-art desktop PC. Typically, thousands of parameter combinations have to be tested, which requires months of CPU time.

By providing seamless access to a number of computing centers world-wide, the EGEE grid has proven useful to support heavy experiments in medical imaging and other biomedical disciplines (see, e.g., [14]).

Within EGEE, resources and users are organized in Virtual Organizations (VO) spanning administrative domains, institutes and countries. The infrastructure aggregates numerous sites, each of them providing a set of worker nodes managed by one or several batch queues published in the information system as Computing Elements (CEs). Jobs are dispatched on CEs by Workload Management Systems (WMS). Each CE may have a local data Storage Element (SE) accessible from any network domain. Users are authentified by X509 certificates and registered in one or several VOs to get access to resources. gLite[1] is the grid middleware operating this infrastructure.

In section 2 Mean-Shift and its grid deployment are described. The experiments and results are then discussed in section 3 highlighting (i) the feasibility of the Mean-Shift scale parameters optimization using a gradient ascent algorithm and (ii) a number of observations on the grid execution.

2. Materials and methods

2.1. Mean-Shift

Mean-Shift theory. Mean-Shift is an algorithm aiming at finding the modes of the estimated density $\hat{f}(\mathbf{x})$. This problem can be written as:

$$\nabla \hat{f}(\mathbf{x}) = 0, \text{ where } \nabla \text{ is the gradient operator.} \tag{1}$$

Using Parzen windowing (a non-parametric kernel-based probability density function estimator), Fukunaga introduced in [1] a gradient ascent algorithm that solves equation (1) iteratively. This iterative process can be expressed as follows:

$$\mathbf{x}^{[t+1]} = \frac{\sum_{i=1}^{n} g\left(d^2\left(\mathbf{x}^{[t]}, \mathbf{x}_i, \mathbf{H}\right)\right).\mathbf{x}_i}{\sum_{i=1}^{n} g\left(d^2\left(\mathbf{x}^{[t]}, \mathbf{x}_i, \mathbf{H}\right)\right)} \tag{2}$$

where n is the number of samples, t is the iteration variable, $d(.,.,\mathbf{H})$ is the Mahalanobis distance with covariance matrix $\mathbf{H}$, $\mathbf{x}^{[0]}$ is set to a given $\mathbf{x}_i$ and $g(.)$ is a weight function deriving from the kernel function. In this work, $g(.)$ is a rectangular function centered on 0, i.e., defining a neighborhood. In this case the kernel for the Parzen windowing is the Epanechnikov kernel, which is the optimal kernel according to the asymptotic mean

[1] `http://glite.web.cern.ch`

integrated square error. The iterative process stops when $d\left(\mathbf{x}^{[t]}, \mathbf{x}^{[t+1]}, \mathbf{H}\right)$ is smaller than a given threshold value. Then $\mathbf{x}^{[t+1]}$ is the corresponding mode of the initial point $\mathbf{x}^{[0]}$. Note that two different approaches exist for the iterative process (see [4]) and that equation (2) corresponds to the non-blurring approach.

Mean-Shift filtering. Mean-Shift image filtering has been firstly described in [10]. Each pixel of the image can be considered as a vector $\mathbf{x}$ in the feature space R^d. Depending on the application, this vector consists of different features like the spatial position, the color components, the normal vector, different weighted MRI images, etc. In this work we use the joint spatial-range domain. Let us denote by $\mathbf{x}_s$ the spatial position and by $\mathbf{x}_r$ the range vector (i.e. the values of pixel $\mathbf{x}_s$), the vector $\mathbf{x}$ can be written as: $\mathbf{x} = \left(\mathbf{x}_s^T, \mathbf{x}_r^T\right)^T$. In our study, the bandwidth matrix $\mathbf{H}$ consists of four scale parameters: h_s (spatial), h_{red}, h_{green} and h_{blue}:

$$\mathbf{H} = \begin{bmatrix} h_s^2 & 0 & 0 & 0 \\ 0 & h_{red}^2 & 0 & 0 \\ 0 & 0 & h_{green}^2 & 0 \\ 0 & 0 & 0 & h_{blue}^2 \end{bmatrix} \tag{3}$$

Note that commonly used scale matrices consist of one or two parameters only. Let $\{\mathbf{x}_i\}_{i=1,2\ldots n}$ and $\{\mathbf{z}_i\}_{i=1,2\ldots n}$ be respectively the original and the filtered image points in the d-dimensional feature space. The mean shift filtering algorithm is summarized as:

- Initialize $\mathbf{x}^{[0]}$ to a given point $\mathbf{x}_i$,
- Using equation 2, compute $\mathbf{x}^{[t+1]}$ until convergence,
- Build $\mathbf{z}_i$ from the spatial position of $\mathbf{x}_i$ and from the range values of $\mathbf{x}^{[t+1]}$, i.e. $\mathbf{z}_i = \left((\mathbf{x}_i)_s^T, \left(\mathbf{x}^{[t+1]}\right)_r^T\right)^T$
- Repeat the above steps for each point $\mathbf{x}_i$ of the image.

The asymptotic time complexity of the basic algorithm is $\theta\left(n^2.\bar{it}\right)$, where n is the number of pixels and $\bar{it}$ is the average number of iterations until convergence. Considering equation (2) with $g(.)$ a rectangular function in both spatial and range domains, the asymptotic time complexity of the used algorithm is $\theta\left(n.m.\bar{it}\right)$, where $m << n$ is the number of pixels in the spatial neighborhood.

Scale parameters optimization. As shown in the results section, scale parameters significantly influence Mean-Shift results quality. For a given $\mathbf{H}$, we assess the filtering quality with the signal-to-noise ratio (P_{SNR}). In order to optimize the choice of $\mathbf{H}$ with a gradient ascent algorithm, the convergence of the gradient ascent to the global maximum of the P_{SNR} must be ensured from all the initial $\mathbf{H}$ values, i.e., there must be only one basin of attraction. This test is done by filtering an image with many different $\mathbf{H}$ matrices, i.e., by sweeping parameters h_s, h_{red}, h_{blue} and h_{green}.

2.2. Grid deployment

General framework. The grid deployment environment is based on the Virtual-Lab for medical imaging described in [15] and inner references. Users interact with the VBrowser with which they can (i) browse and transfer input/output files to/from EGEE grid storage resources and logical file catalog and (ii) parametrize and launch application

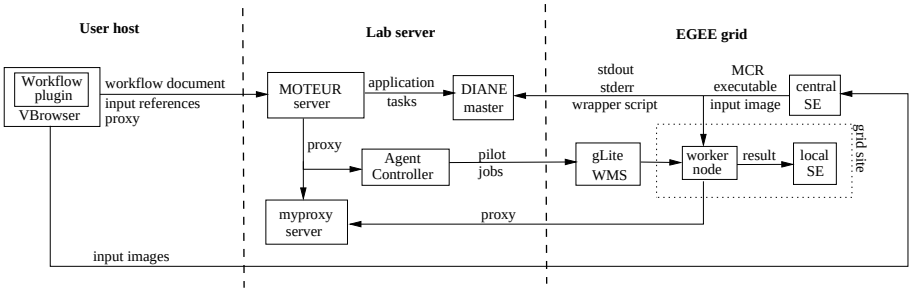


Figure 1. Setup used for the Mean-Shift deployment.

workflows. Workflows are sent with input data references to a server hosted and maintained at the lab. This server sets up basic configuration and starts workflow execution with the MOTEUR engine. As described below, a few add-ons were implemented to improve reliability and performance.

Pilot jobs. The application runs in the EGEE biomed VO composed of some 120 sites each of them having a Storage Element and a few Computing Elements. To improve fault-tolerance and task scheduling on such heterogeneous resources the DIANE pilot-job framework [16] has been integrated. The resulting setup is diagrammed on Figure 1. Tasks generated by the workflow engine are submitted to a master pool hosted at the lab. In parallel, pilot jobs are automatically submitted by an agent controller. To improve reliability, pilots with failed tasks are removed from the master pool.

Application description. The application workflow consists of the main application component (running on the grid), a data transfer component transferring files from the grid to the lab server and a results packing component. Mean-Shift is implemented using proprietary Matlab software[2], which complicates its grid deployment. Licensing issues are tackled by compiling the application code on a machine holding the Matlab license and with the same architecture as the EGEE worker nodes (Linux CentOS-5, 64 bits). The generated binary code can be executed on any host with this architecture provided that it has the Matlab Compiler Runtime (MCR) installed. To avoid cumbersome installation, maintenance and update of the MCR on 120+ VO sites it is downloaded and installed on the fly by application tasks.

Data transfers. Data transfers occur from/to EGEE Storage Elements to/from the worker nodes. Inputs consist of the processed image ($<$ 1 MB), the Matlab Compiler Runtime (127 MB) and a tarball containing the executable and required libraries (1.6 MB). Each application task produces a single file of a couple of megabytes. Additionally a task wrapper is downloaded from the pilot master and standard output and error files are periodically uploaded to the master. Given the number of jobs involved in Mean-Shift experiments (a few thousands) the workflow can be quite data-intensive and it is very likely to disturb performance and reliability of the application. Three measures have been implemented to overcome this issue. First, application jobs are instrumented with a data cache avoiding multiple downloads of the same data file by application tasks suc-

[2] `http://www.mathworks.com/`

cessively running on the same worker node. Second, application jobs are parametrized to upload their results to the site's local Storage Element. A fallback on a central SE is implemented in case the local SE is not properly defined or configured. Finally, a result upload test is performed at the beginning of every job execution. This allows to quickly detect upload problems and to limit their impact in particular in case of very long jobs. This upload test is part of the caching mechanism, i.e., it is performed only once for a given worker node.

Credentials. User credentials remain securely stored on the user's desktop. A grid temporary proxy is initialized from the user certificate *via* the VBrowser GUI. This proxy is delegated to the workflow server before the execution is launched and the pilot master is started. Each user receives her own pilot master, which requires one open port per user in the campus firewall but ensures basic security. Pilot jobs are submitted with user's credentials so that any interaction with grid services (mainly file transfer and job execution) is done in the user's name.

Handling long jobs. Mean-Shift filtering involves jobs with wall-clock times ranging from a few minutes to several dozens of hours. Because VOMS proxy extensions are currently limited to 24 hours, proper proxy renewal is required. Whereas proxy expiration does not impact queued applications tasks (they are locally stored in the pilot master) it kills running tasks, which has a dramatic impact on application performance when task duration is long. To circumvent the issue, the user is asked to periodically renew her proxy and to upload it to the workflow server by submitting a dummy workflow. The renewed proxy is then uploaded to a myproxy[3] server. As first tests with automatic proxy fetching by the gLite WMS proved unreliable a dedicated mechanism was implemented in the tasks generated by the workflow engine. A watch-dog script is started with the application task, it periodically checks proxy duration, it fetches the new proxy from the myproxy server when required and it is killed just after the job finished[4]. Additionally, pilot jobs are submitted to resources with long wall-clock time limit (> 24 hours). Periodical proxy renewals are a bit cumbersome for the users. These could be avoided by asking the user to create a very long proxy and renewing only the VOMS proxy extension, which only requires a valid proxy. The solution would then be completely transparent to the user.

Validation. The grid deployment was prototyped and validated against an execution on a local cluster. The application was compiled and run on two different architectures (Linux Fedora 11 32 bits and 64 bits) and the results were checked to be identical to the ones produced on the grid whatever the architecture.

3. Experiments and results

3.1. Mean-Shift

Experiments description. For this first experiment, a classically used RGB color image is used. Future works will consist in processing different kind of images, including med-

[3] `http://grid.ncsa.illinois.edu/myproxy/`

[4] This script was adapted from Y. Cardenas, CC-IN2P3 Lyon, France.

	$start$	$step$	end
h_s	1	1	5
h_{red}	20	5 to 20	240
h_{green}	20	5 to 20	240
h_{blue}	20	5 to 20	240

Table 1. Scale parameter ranges.

ical data. We use the 512×512 RGB color Lena image as reference image. This image is corrupted by an additive Gaussian noise ($\sigma^2 = 30$). The noisy image is then filtered using the Mean-Shift with different scale matrix. The range of each scale parameters is described in table 1.

For each scale combinations, the P_{SNR}, the computational time t and the average number of iterations $\bar{it}$ is computed.

Results and discussion. More than 20,000 jobs have been completed on the EGEE grid and the local cluster. This study validates the Matlab portability using Mean-Shift work-

(a) (b)

(c) (d)

Figure 2. (a) Original RGB image. (b) Noisy image ($\sigma^2 = 30$). (c-d) Filtered image with 2- and 4-optimal scale parameters.

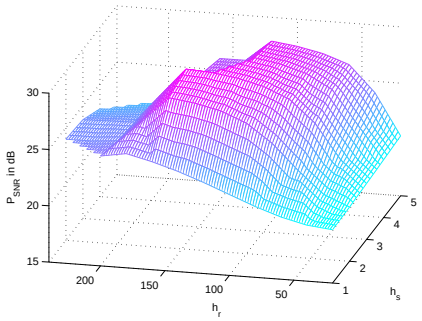


Figure 3. P_{SNR} evolution against h_s and h_r.

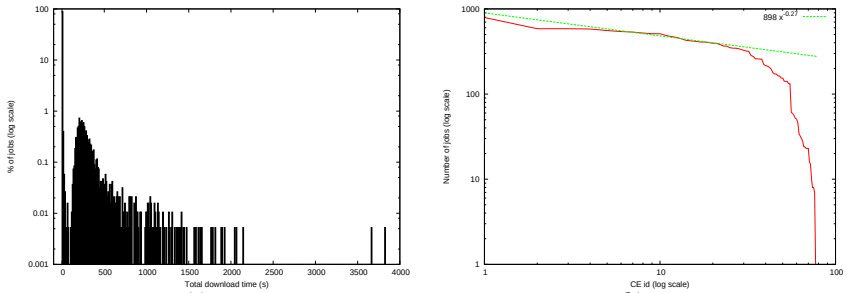
flow on the EGEE grid. Figure 2(a) shows the reference image and Figure 2(b) the noisy image processed by the different scale matrices. The P_{SNR} between these images is 19.71dB. The P_{SNR} evolution against h_s and $h_r = h_{red} = h_{green} = h_{blue}$ is displayed in Figure 3. The P_{SNR} evolves smoothly from 18dB to 28.9dB allowing optimization by a gradient ascent algorithm. The best result obtained with the 2-parameter matrix is $P_{SNR} \approx 28.93dB$ for $h_s = 3$ and $h_r = 160$. The optimal values for the 4-optimal scale parameters are $h_s = 3$, $h_{red} = 240$, $h_{green} = 140$, $h_{blue} = 200$ and the $P_{SNR} \approx 29.01dB$. The obtained images are displayed in 2(c) and 2(d). The differences between 2- and 4-scale parameters involve few pixels highly corrupted by the noise. This approach shows that the optimization of two parameters can be sufficient in most cases. Nevertheless, if all the four parameters are needed, it is easier to optimize them starting from the previous two-parameters optimization. Using a neighborhood of radius N, we search the basins of attraction for the 4 dimensional space of scale parameters. For $N = 1.2$, 21 basins are obtained (from 23.98dB to 29.01dB). For $N = 2$ ($= \sqrt{4}$), 3 basins are obtained (from 28.98dB to 29.01dB). For $N = 2.5$, only one basin is obtained (29.01dB). From this experiment, using a gradient ascent algorithm with $N \geq 2$ in order to optimize scale parameters is possible and gives quasi-optimal scale parameters.

3.2. Grid facts

Experiment description. The grid experiment consisted in the execution of 19,000 Mean-Shift jobs all processing the same image. For scalability reasons, the experiment was split in 5 batches corresponding to different values of parameter h_s (varying from 1 to 5). Batches $h_s = 1$ and $h_s = 2$ were composed of 8,000 Mean-Shift jobs while the other ones had only 1,000.

h_s value	Total CPU time	Elapsed time	Speed-up	Total data transfer time	Produced data	Successful tasks	Total tasks	Error ratio
1	13.0 days	3h25min	92	89.8h	43GB	8,000	8,106	1.3%
2	50.4 days	18h36min	65	63.2h	41GB	8,000	8,929	10.4%
3	17.3 days	13h01min	32	34.8h	5GB	1,000	1,317	24%
4	29.0 days	13h23min	52	44.7h	5GB	1,000	1,089	8.2%
5	54.3 days	11h09min	117	51.9h	5.1GB	1,000	1,179	15.2%

Table 2. Grid performance.



(a) (b)

Figure 4. (a) Histogram of input download times. Data caching yields a very good improvement (see peak at 0s). (b) Total number of pilot jobs received by CE. CEs are ordered by decreasing order of their received number of pilot jobs. The regression is done on the first 30 CEs and suggests an exponential decrease.

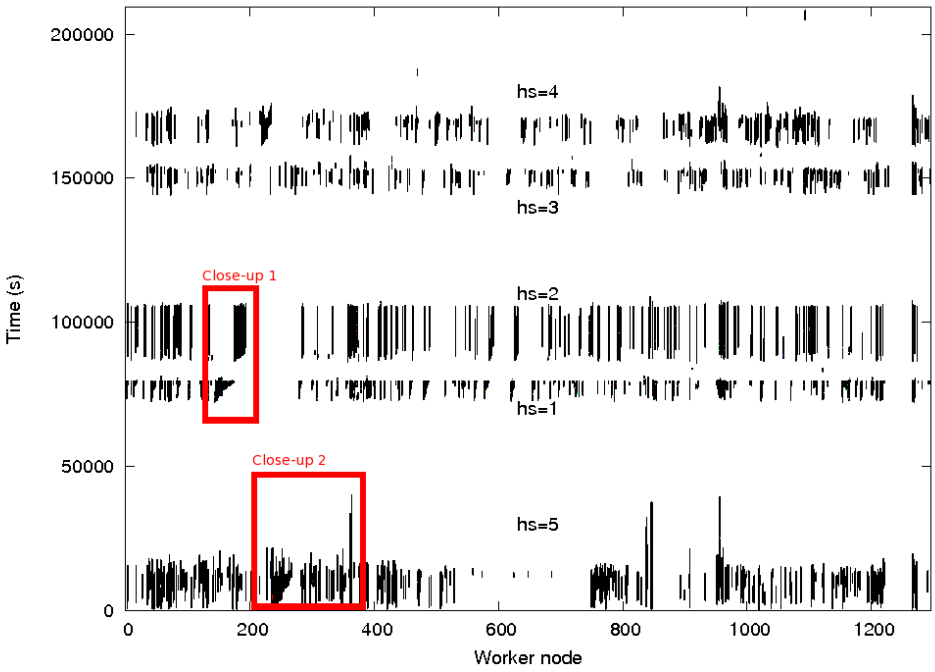
Achieved performance. Table 2 shows the performance of the 5 job batches. Overall the EGEE grid allowed to compute a 164-day experiment in 2.5 days, i.e., it sped up the experiment by a factor of 66. In spite of the use of pilot jobs, error ratios are quite important (7.8% in average) as also reported on other production grids [17]. Data transfers account for a large majority (86.6%) of the errors while the other error causes are communication problems between tasks and the pilot master (8.8%) and application errors due to misfunctionings of the Matlab Compiler Runtime on worker nodes (4.8%). The error ratio slightly varies among parameter sets but no correlation is observed. The error ratio for hs=3 is particularly high (24%) due to a lot of data transfer failures in this batch. This may be due to a temporary downtime of some SEs. Eventually all the application tasks were computed but at the cost of resubmissions.

Input download. The distribution of input download times computed on the 19,000 jobs is shown on Figure 4(a). Thanks to data caching, 88% of the jobs had a total download time inferior to 10 seconds. Extreme download times (> 1500s) are observed. The corresponding 16 jobs ran on 4 sites that are probably poorly connected to the central SE hosting the input data.

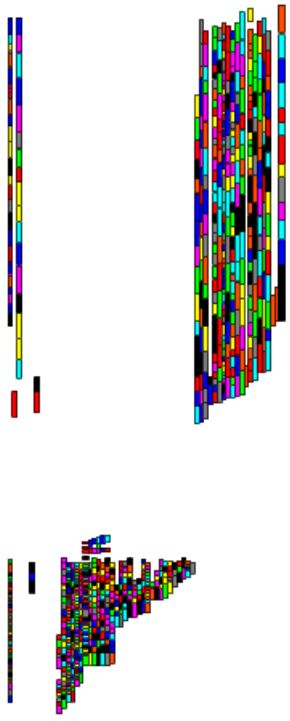
Result upload. 23% of the 19,000 result files could not be transferred to the computing site's local SE and thus were uploaded to the central. This concerned 12 sites and came from various problems (1 full SE, 4 authentication issues and 7 configuration problems). Overall the average result transfer time was reduced from 24.4 seconds using the central SE to 14.4 seconds using the local SEs.

Pilot job scheduling. Two different scheduling processes are involved in this experiment. Scheduling of pilot jobs on grid sites is done by the glite WMS and scheduling of application tasks on available pilots is done by the pilot master. In this experiment pilots ran on a total number of 1,295 worker nodes managed by 79 batch queues belonging to 66 grid sites. Similarly to what is observed in [18] (see Figure 3 inside), Figure 4(b) suggests that the distribution of the number of (pilot) jobs received per Computing Element is quite accurately approximated by an exponential function.

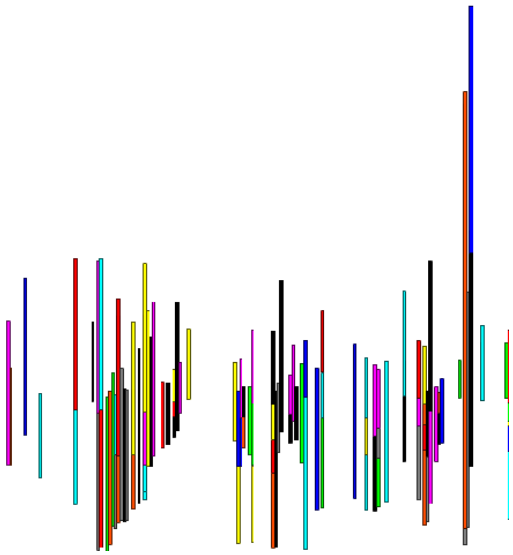
Task scheduling. Figure 5 plots the Gantt diagram of the successful application tasks. The five experiment batches (one per h_s value) are clearly visible. The scheduling of short-task batches seems reasonable: in batches $h_s = 1$ and $h_s = 2$ worker nodes are



(a) Gantt diagram of the experiment

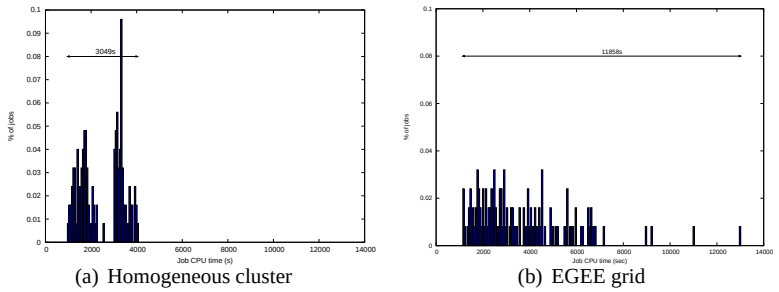


(b) Close-up 1 (hs=1 and hs=2 ; short tasks)



(c) Close-up 2 (hs=5, long tasks)

Figure 5. Gantt diagram of the successful application tasks. Pilots are identified by the worker node where they are running and are reported on the x axis. The y axis represents time and covers the whole range of the experiment (from the launch of the first batch to the completion of the last task of the last batch). Tasks are figured by bars centered on the worker node id and covering the time interval in which they ran on this resource (including execution time and data transfers). Worker nodes are ordered by increasing performance (measured by the Bogomips value).



(a) Homogeneous cluster (b) EGEE grid

Figure 6. CPU time histograms on a homogeneous cluster (a) and on the EGEE grid (b) for hs=5 and h_{red}, h_{green} and h_{blue} in 40,80,120,160,200 (125 jobs on each infrastructure).

released at instants close to each other, which indicates good resource exploitation. On the contrary, the scheduling of long-task batches is clearly an issue. Resource release times are very heterogeneous and it has an important impact on the makespan. For $h_s = 5$, removing the 7 last tasks would reduce the makespan by a factor of 2. Solutions to cope with this "last task issue" as studied in [19] should be considered in the future.

Comparison with cluster execution. A smaller-scale experiment was launched on a homogeneous cluster of six 8-core nodes (Intel Xeon L5430 2.66 GHz) managed by PBS. Mean-Shift filtering was performed with hs=5 and 125 combinations of h_{red}, h_{green} and h_{blue}. CPU time histograms are reported on Figure 6. The total CPU time is 1.5 times higher on the grid than on the cluster (126.3h vs 85.8h), which is in the order of magnitude of the average Bogomips ratio (5320 for the cluster nodes VS 4637 for the grid nodes, i.e., a 1.14 ratio). As shown on the graphs the variability is multiplied by a factor of almost 4, which may be an explanation for scheduling issues noticed above.

Usability. Although the experiment was prepared and launched autonomously by the end-user after a brief hands-on of the environment, a significant assistance from grid engineers was required, mainly for application porting, data retrieval and experiment planning. As discussed above, using local Storage Elements reduces the average results upload time. But it also spreads the results on 50+ sites, which increases the chance to face an SE downtime during results transfer and packing. Some transfers have to be retried a number of times, which is not automated yet and requires knowledge of the grid data clients. Besides, assistance in experiment planning was required to split the 19,000 jobs of the experiment in reasonable batches (a single 19,000-job batch would probably not have been possible for scalability reasons) and to react to operational issues. Moreover, in spite of high-level porting tools (workflow managers and software wrappers), application porting was done by grid engineers who took care of technical issues related to debugging and fine-tuning of the application to adjust the grid conditions.

4. Conclusion

Mean-Shift filtering quality is highly dependent on the scale parameters. This study showed that optimizing the scale parameters using a gradient ascent algorithm gives quasi-optimal results. Optimization of 2 parameters leads to optimal solution. For the 4 parameters optimization, large neighborhoods (bigger than $\sqrt{4}$) should be considered in

order to converge to the optimal solution. In future works this framework will be applied to various images of the literature and to medical data. We expect to understand and find a robust link between optimal scale parameters and data content or acquisition system.

The EGEE grid allowed to compute a 164-day experiment in 2.5 days, i.e., it sped up the experiment by a factor of 66. Despite the use of pilot jobs, the reliability remains quite poor, with an average task error ratio of 7.8%, mainly coming from data transfers. Anyhow all application tasks were eventually computed thanks to resubmissions. Data caching and the use of local Storage Elements significantly reduced the total transfer time. In addition, result analysis shows that task scheduling of long tasks could be improved and that pilot job dispatching seems to be fairly fitted by an exponential distribution, as suggested by other works.

Overall the experiment described in this paper has shown that the EGEE grid and associated application-level middleware was exploitable by end-users as a scientific instrument to design and run imaging experiments. The execution environment is mature enough to envisage production exploitation by a number of other applications in the medical imaging domain. Still, engineering assistance was required for application porting, experiment planning and result retrieval. Future works will concentrate on further increasing the robustness and performance of the framework to give users full autonomy with grid experiments in their daily scientific routine.

Acknowledgments

This work is partially funded by the European Commission through the EGEE-III project, contract number INFSO-RI-22266. It is also supported by CNRS "Institut des Grilles". The authors would like to thank the numerous teams behind the GGUS for their support and Fabrice Bellet for his technical help.

References

[1] K. Fukunaga and L. D. Hostetler. Estimation of the gradient of a density function with applications in pattern recognition. *IEEE Transaction on Information Theory*, 21(1):32–40, 1975.

[2] D. Comaniciu and P. Meer. Robust analysis of feature spaces: Color image segmentation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 750–755, San Juan, PR, USA, 1997.

[3] O. Wirjadi and T. Breuel. A branch and bound algorithm for finding the modes in kernel density estimates. *International Journal of Computational Intelligence and Applications*, 8(1):17 – 35, 2009.

[4] Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE Transaction Pattern Analysis and Machine Intelligence*, 17(8):790–799, 1995.

[5] M. Fashing and C. Tomasi. Mean shift is a bound optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):471–474, 2005.

[6] D. Barash and D. Comaniciu. A common framework for nonlinear diffusion, adaptive smoothing, bilateral filtering and mean shift. *Image and Vision Computing*, 22(1):73–81, 2004.

[7] H. Tek, D. Comaniciu, and J.-P. Williams. Vessel detection by mean shift based ray propagation. In *Workshop on Mathematical Methods in Biomedical Image Analysis*, pages 228–235, Kauai, HI, 2001.

[8] T. Grenier, C. Revol-Muller, F. Davignon, O. Basset, and G. Gimenez. Multiparametric smoothing based on mean shift procedure for ultrasound data segmentation. In *Proceedings of the EUSIPCO'05*, pages Article ID cr1461, 4 pages, Antalya, Turkey, 2005.

[9] T. Grenier, C. Revol-Muller, and G. Gimenez. Hybrid approach for multiparametric mean shift filtering. In *IEEE International Conference on Image Processing*, pages 1541–1544, Atlanta, USA, 2006.

[10] D. Comaniciu and P. Meer. Mean shift analysis and applications. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2, pages 1197–1203, Kerkyra, Greece, 1999.

[11] D. Comaniciu, V. Ramesh, and P. Meer. The variable bandwidth mean shift and data-driven scale selection. In *International Conference on Computer Vision*, volume 1, pages 438–445, Vancouver, 2001.

[12] D. Comaniciu. An algorithm for data-driven bandwidth selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):281–288, 2003.

[13] T. Grenier, C. Revol-Muller, F. Davignon, O. Basset, and G. Gimenez. Variable bandwidth mean shift for smoothing ultrasonic images. In *Proceedings of the EUSIPCO'05*, pages Article ID cr1454, 4 pages, Antalya, Turkey, 2005.

[14] Nicolas Jacq, Jean Salzeman, Florence Jacq, Yannick Legré, Emmanuel Medernach, Johan Montagnat, J. Maass, M. Reichstadt, H. Schwichtenberg, M. Sridhar, V. Kasam, M. Zimmermann, M. Hofmann, and Vincent Breton. Grid-enabled Virtual Screening against malaria. *Journal of Grid Computing (JOGC)*, 6(1):29–43, March 2008.

[15] Silvia Olabarriaga, Tristan Glatard, and Piter T. de Boer. A virtual laboratory for medical image analysis. *IEEE Transactions on Information Technology In Biomedicine (TITB)*, pages (in–press), 2010.

[16] Jakub T. Mościcki. Distributed analysis environment for HEP and interdisciplinary applications. *Nuclear Instruments and Methods in Physics Research A*, 502:426 – 429, 2003.

[17] Dagmar Krefting, Ralf Luetzkendorf, Peter Kathrin, and Johannes Bernarding. Performance Analysis of Diffusion Tensor Imaging in an Academic Production Grid. In *CCGrid-Health'10*, May 2010.

[18] Lovro Ilijasic and Lorenza Saitta. Characterization of a Computational Grid as a Complex System. In *Grid Meets Autonomic Computing(GMAC'09)*, pages 9–18, June 2009.

[19] W Cirne, F. Brasileiro, D. Paranhos, L.F.W. Goes, and W. Voorsluys. On the efficacy, efficiency and emergent behavior of task replication in large distributed systems. *Parallel Computing*, 33:213–234, 2007.